Variations on Neighborhoods in CA(Theory of Computer Science and Its Applications)

AUTHOR(S):

Worsch, Thomas; Nishio, Hidenosuke

CITATION:

Worsch, Thomas ...[et al]. Variations on Neighborhoods in CA(Theory of Computer Science and Its Applications). 数理解析研究所講究録 2007, 1554: 24-31

ISSUE DATE:

2007-05

URL:

http://hdl.handle.net/2433/80976

数理解析研究所講究録
第1554巻 2007年 24-31

Variations on Neighborhoods in CA

トーマス・ヴォルシュ（カールスルーエ大情報）
Thomas Worsch (University of Karlsruhe, Germany) [1]
email : worsch@ira.uka.de

and

西尾英之助（元・京大理）
Hidenosuke Nishio (ex. Kyoto University)
email: YRA05762@nifty.com

Abstract

We show how an arbitrary finite number of CA with different local rules can be simulated by CA using *the same local rule* by just changing the (shape of the) neighborhood. In that way one can even achieve universality.

1 Introduction

Usually investigations of cellular automata without further discussion assume some standard neighborhood because it is "without loss of generality". In general this is correct, except, of course, when one is interested in questions specifically concerning neighborhoods.

The rest of this paper is organized as follows: In Section 2 we introduce some basic notation we are going to use later on. The following two sections are devoted to simulations of some CA $\mathcal{A}_i$ by other CA, where the latter only differ in their neighborhoods.

Sections 3 presents a solution to the conceptually simpler task of simulating a finite number of CA $\mathcal{A}_i$ which all have the same neighborhood but different local functions by CA $\mathcal{B}_i$ which all have the same local function but different neighborhoods.

In Section 4 it will be shown that one can even achieve universality in the following sense: There is *one* local rule which is used by different CA with different neighborhoods in such a way that any CA $\mathcal{A}_i$ with state set $\{0,1\}$ for its cells can be simulated, even (and in particular) if the initial configuration does not contain any information about the CA to be simulated. That does only depend on the specific neighborhood of the simulating CA.

Throughout the paper we prefer solutions that can be described and used easily over solutions which are optimized for running time or the number of states per cell.

2 Basics

We will describe the construction in Section 3 for d-dimensional CA and denote by $R = \mathbb{Z}^d$ the set of all cells (in Section 4 we will restrict ourselves to the case $d = 1$). Let Q denote the finite set of states for each cell and N the finite neighborhood containing $n = |N|$ relative offsets to cells. Without loss of generality we assume that $\mathbf{0} = (0,\ldots,0) \in N$. Let $\nu : \{0,\ldots,n-1\} \to N$ be a bijection satisfying $\nu(0) = \mathbf{0}$. This function is introduced in order to have a numbering of the neighbors. It allows to use "the same local rule" with different neighborhoods. Equivalently one can think of ν as a vector $(\nu(0),\ldots,\nu(n-1))$.

Global configurations are formalized as mappings $c : R \to Q$; thus $c(j)$ is the state of cell j in configuration c. The local rule $f : Q^n \to Q$ induces the global function in the usual way. If the CA is

[1] corresponding author

$\mathcal{A} = (R, Q, N, \nu, f)$ we write $\mathcal{A} : Q^R \to Q^R$ for the global function.

$$\mathcal{A}(c)(j) = f(c(j + \nu(0)), c(j + \nu(1)), \ldots, c(j + \nu(n-1))) .$$

3 Simulating several CA using one local function: a simple idea

For $0 \leq i < m$ let $\mathcal{A}_i = (R, Q_A, N_A, \nu_A, f_i)$ denote m cellular automata. Configurations $c : R \to Q_A$ are simply called $\mathcal{A}$-configurations since all these CA have the same set of configurations.

We will describe m CA $\mathcal{B}_i = (R, Q_B, N_i, \nu_i, f_B)$ such that each $\mathcal{B}_i$ simulates $\mathcal{A}_i$ (in an obvious sense). Configurations $c : R \to Q_B$ are simply called $\mathcal{B}$-configurations since all these CA have the same set of configurations.

Note that *the simulating CA $\mathcal{B}_i$ use the same set of states and the same local function and they only differ in (the shapes of) their neighborhoods.* Furthermore the same embedding of $\mathcal{A}$-configurations into $\mathcal{B}$-configurations will be used for all simulations.

We use $Q_B = Q_A \times \{0, 1, \ldots, m-1\}$. For any $q \in Q_B$ we write $s(q)$ and $p(q)$ for the first and second component of q respectively.

The following embedding $E : Q_A^R \to Q_B^R$ of configurations will be used:

$$E(c)(j) = (c(j), j[0] \bmod m)$$

where $j[0]$ means the first component of the vector j.

For example, if we want to simulate $m = 3$ one-dimensional CA $\mathcal{A}_i$, then $\mathcal{A}$-configurations like

$\cdots$	q_{-5}	q_{-4}	q_{-3}	q_{-2}	q_{-1}	q_0	q_1	q_2	q_3	q_4	q_5	$\cdots$

will be embedded into

$\cdots$	q_{-5}	q_{-4}	q_{-3}	q_{-2}	q_{-1}	q_0	q_1	q_2	q_3	q_4	q_5	$\cdots$
$\cdots$	1	2	0	1	2	0	1	2	0	1	2	$\cdots$

Lemma 1 *Let c_A be an arbitrary $\mathcal{A}$-configuration and $c_B = E(c_A)$. For all $i, j \in R$ and all offsets $x \in \mathbf{N} \times \{0\}^{d-1}$ holds:*

$$p(c_B(i + x)) - p(c_B(i)) = p(c_B(j + x)) - p(c_B(j)) \pmod m$$

This should be obvious since the equation is equivalent to

$$p(c_B(i + x)) - p(c_B(j + x)) = p(c_B(i)) - p(c_B(j)) \pmod m .$$

Definition 2 *The CA $\mathcal{B}_i$ are specified as follows:*

- *$N_i = N \cup \{r_i\}$ where $r_i = (r_i, 0, \ldots, 0)$ and r_i is the smallest positive integer not occurring in any offset $n \in N$ as a component and $r_i = i \bmod m$.*
- *The numbering of neighbors is basically the same as for the $\mathcal{A}_i$:*

$$\nu_i(j) = \begin{cases} \nu_A(j) & \text{iff } j < n \\ r_i & \text{iff } j = n \end{cases}$$

- *We define $f_B : Q_B^{n+1} \to Q_B$ by specifying the two resulting components separately:*

$$\begin{aligned} s(f_B(q_0, \ldots, q_n)) &= f_i(s(q_0), \ldots, s(q_{n-1})) \\ &\quad \text{where } i = s(q_n) - s(q_0) \bmod m \\ p(f_B(q_0, \ldots, q_n)) &= p(q_0) \end{aligned}$$

Lemma 3 *$\mathcal{B}_i$ simulates $\mathcal{A}_i$ (for all $0 \leq i < m$) in the following sense: For any $\mathcal{A}$-configuration c and all $t \geq 0$ one has*

$$E(\mathcal{A}_i^t(c)) = \mathcal{B}_i^t(E(c))$$

Proof: Consider an arbitrary i and an arbitrary $\mathcal{A}$-configuration c. The case $t = 0$ is trivial. Hence it suffices to prove the claim for $t = 1$, the rest follows by an easy induction.

From the definition of $p(f_B(\ldots))$ immediately follows that the second component of a $\mathcal{B}$-cell does never change its value (since we always assume $\nu(0) = 0$).

It remains to have a look at the first components. Consider an arbitrary cell $j \in R$. By definition of E one gets

$$s(E(\mathcal{A}_i(c))(j)) = \mathcal{A}_i(c)(j) = f_i(c(j + \nu_A(0)), \ldots, c(j + \nu_A(n-1)))$$

On the other hand for $c' = E(c)$ holds

$$\begin{aligned} s(\mathcal{B}_i(c')(j)) &= s(f_B(c'(j + \nu_i(0)), \ldots, c'(j + \nu_i(n-1)), c'(j + \nu_i(n)))) \\ &= f_x(s(c'(j + \nu_i(0))), \ldots, s(c'(j + \nu_i(n-1)))) \\ &= f_x(s(c'(j + \nu_A(0))), \ldots, s(c'(j + \nu_A(n-1)))) \\ &= f_x(c(j + \nu_A(0)), \ldots, c(j + \nu_A(n-1))) \end{aligned}$$

By Lemma 1 the value x is the same for all cells j and by the definition of $s(f_B(q_0, \ldots, q_n))$ it is $x = s(q_n) - s(q_0) \bmod m = r_i - 0 \bmod m = i$.

Hence one gets exactly the same value as on the left hand side of the claim. ■

4 Simulating several CA using one local function: a universal solution

In order to make it easier to describe the construction, only one-dimensional CA will be considered from now on. The generalization to the higher dimensional case is at most tedious but not difficult.

In this section we will describe CA $\mathcal{U}_i$ which can simulate any CA $\mathcal{A}_i$ having 2 states per cell, a neighborhood of arbitrary size and shape and an arbitrary local function. As in Section 3, all $\mathcal{U}_i$ will only differ in the shape of their neighborhoods. The main difference to the previous construction is, that, in order to achieve universality, we now have to deal with *infinitely many* $\mathcal{A}_i$.

Also, the embedding of $\mathcal{A}$-configurations into $\mathcal{U}$-configurations will be different from the one in Section 3. But, of course, we will of maintain the feature that the embedding is independent of the specific CA $\mathcal{A}_i$ to be simulated. We choose $Q_U = \{0, 1\} \times Q'$ for some set Q' which contains the symbols $\circ$ and $\llcorner$ (among others). The following embedding $E : Q_A^R \to Q_U^R$ of configurations will be used:

$$E(c)(j) = \begin{cases} (c(j), \circ) & \text{if } j = 0 \\ (c(j), \llcorner) & \text{otherwise} \end{cases}$$

I.e., nothing is changed, except that a special marker $\circ$ is set in one cell. It doesn't matter which cell; we have chosen $j = 0$.

Conceptually, the work of any $\mathcal{U}_i$ consists of three phases:

1. A representation of the CA $\mathcal{A}_i$ to be simulated is generated as a binary string.
2. The input representing the initial configuration for $\mathcal{A}_i$, is transformed.
3. $\mathcal{A}_i$ is simulated by $\mathcal{U}_i$.

As will be seen later on, the second and third phase are overlapped. In the following subsections we will sketch the most important aspects of the construction:

1. how the number of the CA to be simulated is generated as a binary string;
2. how the CA to be simulated is represented in this string;
3. how the initial configuration is transformed for simulation;
4. how one step of one cell can be simulated;

4.1 Computing the binary string representing the CA to be simulated

Consider the following CA: The set of states[2] is $Q_1 = \{0,1,␣\} \times \{0,1,␣\} \times \{\bullet, \circ, >, <, ␣\}$. We will say that the cells consist of three *registers* containing "*sub-states*". In diagrams sub-states ␣ will not be shown; instead corresponding registers will simply be left empty and called *empty*. The first registers (shown at the top of each cell in Figure 1) are used for "sum bits", the second (shown in the middle) are used for "carry bits" and the third (shown at the bottom) are used for signals.

The neighborhood is $N = \{-r, -1, 0, 1, r\}$ where r is an arbitrary number greater or equal to 2. The neighbor at position r is called the *"remote" neighbor* of the origin. It can identify itself, because it sees the marker $\circ$ at its neighbor at position $-r$.

We will explain the local rule with the help of Figure 1. The CA will be started in a configuration where all parts of all cells are empty except one cell. Without loss of generality assume that this is cell 0 and call it the *origin* cell from now on. Initially its first two registers are empty, while the third contains $\circ$.

The goal is to reach a configuration which contains the number r, i.e. the distance of the remote neighbor, in binary representation in the first registers of some cells.

To achieve this the CA does the following in parallel:

- Using the third registers a signal (depicted > in Fig. 1) is started from the origin to the right. It is passed on to the next neighbor until it reaches the cell *which observes the state of the origin cell as its neighbor at* $-r$. There the signal reverses its direction (depicted <) and moves back to the origin where it changes $\circ$ to $\bullet$ after $2r$ steps indicating that everything has been done.
- Simultaneously to the left of the origin a counter is established, initialized with 0 and incremented in each step until the origin observes that the signal just described has arrived at its remote neighbor. Thus the final counter value is r. The time until the signal comes back to the origin is used to process the 1 carry bits in the counter cells if there are any.

4.2 Representation of a CA to be simulated

It is clear that each two-state CA can be represented as a word over some fixed alphabet. Without going into details let us simply give an example: As the alphabet one could use $\{0, 1, +, -, \pm, [,]\}$.

As the complete encoding of a CA one can use the concatenation of the encodings of the neighborhood and the encoding of the local transition functions. For example, the neighborhood $\nu = (-2, 0, 1)$ can be encoded as

```
[[-10] [±0] [+1]]
```

For example, the local transition function of rule 90 can be encoded as

```
[[0000] [0011] [0101] [0111] [1000] [1011] [1101] [1110]]
```

[2] Basically, this set Q_1 is part of the set Q' mentioned in the beginning of this section.

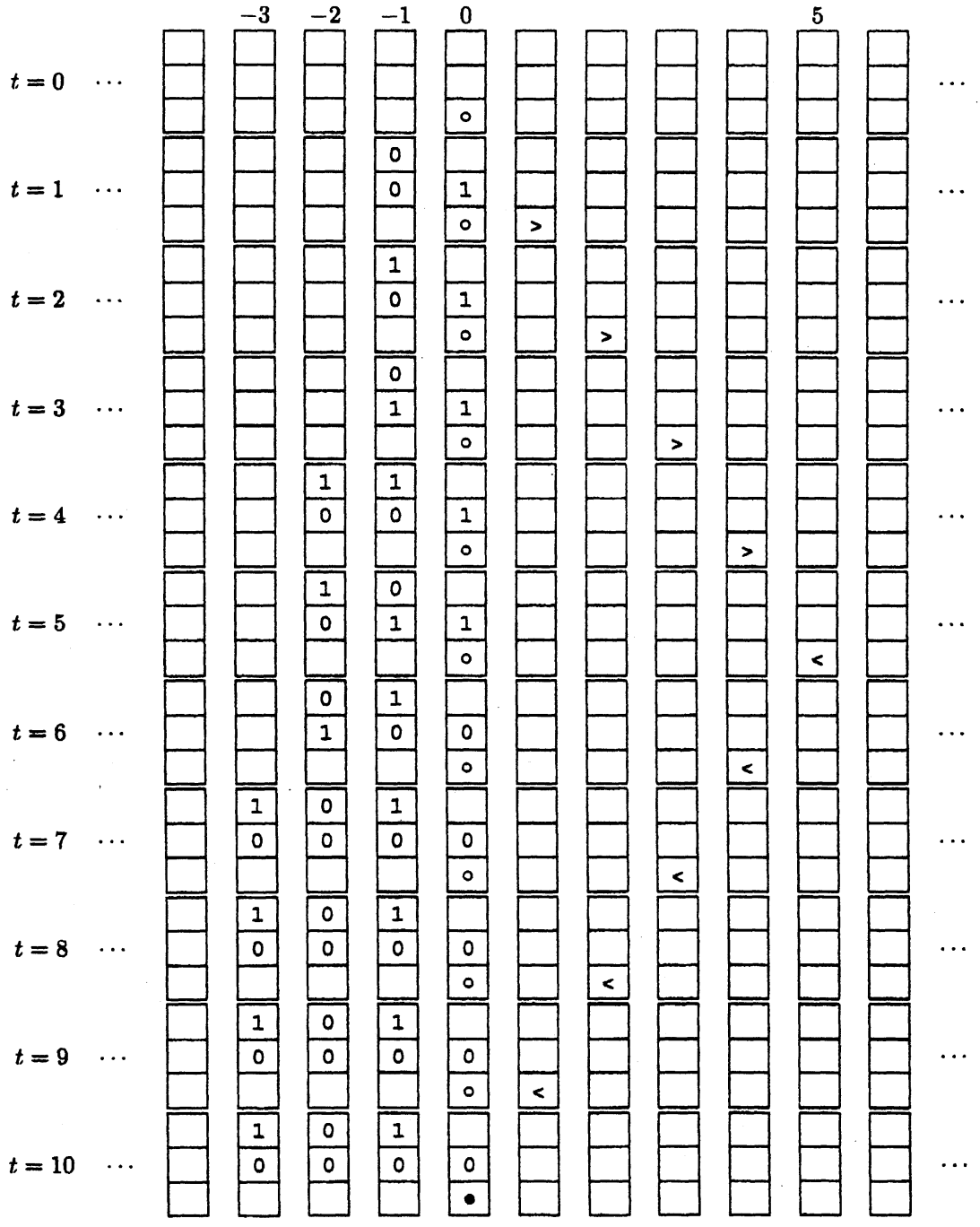

Figure 1: A CA converting the distance of the remote neighbor to a binary number ($r = 5$).

We call a sub-word of the form $[q_0, \ldots, q_{n-1}, f(q_0, \ldots, q_{n-1})]$ a *local rule*. If the i-th neighbor of a cell is in state s, we say that $[q_0, \ldots, q_{n-1}, f(q_0, \ldots, q_{n-1})]$ is *relevant*, iff $q_i = s$. For example, if the state of the second neighbor is 0, the first, second, fifth and sixth local rule are relevant.

The CA $\mathcal{U}_i$ produce binary strings as representations. For that the symbols above have to be encoded as bit strings of some fixed length. It is clear that a CA can check whether an arbitrary binary string really is the encoding of a CA as described above quite easily.

4.3 Transforming the configuration to be used for the simulation

For the simulation each $\mathcal{U}$-configuration consists of *blocks*. Each block

- contains a complete copy of the description of $\mathcal{A}_i$ as explained in the previous subsection,
- is responsible for the simulation of one $\mathcal{A}_i$-cell,
- stores the current and the previous state of the represented $\mathcal{A}_i$-cell, and
- together with each state one of the integers $\{0, 1, 2\}$. They are used for Nakamura's technique [3] of simulating synchronous updating in a framework where the updates cannot really be synchronous.

After the first phase, $\mathcal{U}_i$ has one copy of the description of $\mathcal{A}_i$. The cells containing this copy form the block for simulating cell -1 of $\mathcal{A}_i$; we call this the "block -1". In the sequel $\mathcal{U}_i$ has to

- establish more and more subsequent blocks to the left and to the right with copies of the description of $\mathcal{A}_i$,
- shift the states of the $\mathcal{A}_i$ cells $-2, -3, -4, \ldots$ to the left and the states of the $\mathcal{A}_i$-cells $0, 1, 2, \ldots$ to the right to their corresponding blocks, and
- initialize the mod 3 counters.

Since there is an infinite number of cells to be simulated, this process will never come to an end. But after a finite number of steps enough blocks will be set up so that the first step of cell -1 can be simulated. At some later time, there will be enough blocks to simulate the first step of each neighbor of cell -1. Hence afterwards the second step of cell -1 can be simulated, etc.

We assume that the reader is familiar enough with standard CA techniques, so that she/he could fill out the details here.

4.4 The simulation

Whenever a block wants to simulate one state transition of the represented $\mathcal{A}_i$-cell, it does the following:

1. It sends some kind of state request signals to the "neighbor blocks", i.e., the blocks representing the neighbors of the cell to be simulated.
2. If such a block has already made enough simulations steps (according to Nakamura's technique), it can attach the needed state to the signal and send it back to its originating block. Otherwise the information is sent back, that the state is not yet available.
3. Upon arrival of a state in the block that had requested it, the corresponding rules of the transition table are marked as relevant. If a "not-yet-available" value returns, the request signal is sent again.

4. If all signals have returned a valid state, the new state can be read off the transition table and stored as the new current state while moving the old the previous register. The mod3 counter of the new current state gets 1 plus the value of the now previous state.

The two basic technical aspects which deserve further explanation are the signals and how they are used to select the correct "row of the transition table".

The signals have to know how many blocks the have to travel (and they have to travel as many blocks back to their origin). One can use a standard signal of constant speed (strictly smaller than 1 for the algorithms described below to work) and attach to it a pair of binary numbers (d, D), which initially are both the number of blocks the signal has to travel. While d is used several times for counting down to zero, D is never changed and used to restore the original value for d.

The binary values can be arbitrarily large and hence have to be stored in a distributed fashion in some subsequent cells. When travelling to the neighbor block, d is decremented at each left (or right, depending on the direction of travel) block boundary it passes. The signal has reached its destination block when the value has become 0. For the travel back d is reset to the initial value from D.

The selection of the relevant local rule, it is most convenient, to image that the signals just described have another pair of numbers (k, K) attached to them (in addition to d and D). It is the index of the neighbor: If initially $d = D = \nu(i)$, then $k = K = i$. Analogously to the above, K is used to restore the original value of k, whenever k has been decremented to 0. This is done during the process of marking relevant local rules:

Whenever a signal is returning from the K-th neighbor with some state s, it passes all local rules $[q_0, \ldots, q_{n-1}, f(q_0, \ldots, q_{n-1})]$. Each time it arrives at some q_i, it checks the content of k.

- If $k = 0$, q_i is compared with s and if they are identical, q_i is marked as relevant.
- If $k > 0$, the value k is decremented and the signal moves on to q_{i+1}.

At the end] of the local rule, k is restored to K, so that the next local rule can be checked analogously.

In addition, each such signal can always check, whether *all* states $q_0, \ldots, q_{n-1}$ are marked as relevant. If this is the case, the state at the end of the local rule (immediately before the closing]) is the new state of the simulated cell.

5 Summary and outlook

In Section 3 we he have shown that it is possible to simulate a finite number of CA $\mathcal{A}_i$ with the same neighborhood and different local rules by CA $\mathcal{B}_i$ with the same local rule and different neighborhoods.

In Section 4 we he have shown that it is even possible to simulate an infinite number of CA $\mathcal{A}_i$ with neighborhoods of different size and shape and different local rules by CA $\mathcal{U}_i$ with the same local rule and different neighborhoods which all have size 5.

In both constructions the embedding of the configurations of the simulated CA into configurations of the simulating CA were independent of the specific CA to be simulated. It was *only* the difference in the neighborhoods of the simulating CA that could be and was exploited.

The two constructions did make use of different types of embedding of configurations and different types of simulations. It remains to be investigated, whether in the construction showing universality one really needs to "break symmetry" by setting the special marker in one cell.

We just want to point out that it would not be the first such case. E.g., when looking at the simulation of arbitrary CA by reversible CA for infinite configurations, it is known that this is possible for some kind of simulation [1] while it is provably impossible for another type of simulation [2].

References

[1] Jérôme O. Durand-Lose. Reversible space-time simulation of cellular automata. *Theoretical Computer Science*, 246:117–129, 2000.

[2] Peter Hertling. Embedding cellular automata into reversible ones. In C. S. Calude, J. Casti, and M. J. Dinneen, editors, *Unconventional Models of Computation*, pages 243–256. Springer-Verlag, 1998.

[3] Katsuhiko Nakamura. Synchronous to asynchronous transformation of polyautomata. *Journal of Computer and System Sciences*, 23:22–37, 1981.

manuscript for "kokyuroku" of LA symposium at RIMS, Jan. 2007, March 29, 2007